Advanced Enterprise Objects

Database Access with WebObjects

Session 616

Advanced Enterprise Objects

Database Access with WebObjects

Ben Trumbull
Steve Miner
Sr. Engineers, Development Technologies
webobjects@apple.com

Introduction

Getting more from enterprise objects framework

- WebObjects 5.2 changes
- Advanced topics overview
- Community resources

What this Session is not

WebObjects Developer Documentation

http://developer.apple.com/techpubs/webobjects

WebObjects 5.2 Delta Documentation { READ ME! }

http://developer.apple.com/techpubs/webobjects/Whatsnew

EOF Developer's Guide { READ ME! }

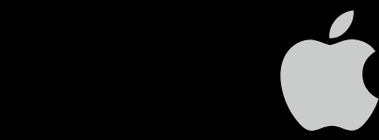
http://developer.apple.com/techpubs/webobjects/Enterprise_Objects

WebObjects API Reference

http://developer.apple.com/techpubs/Reference/API

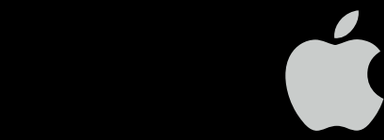
What You'll Learn About

- Memory management
- Data freshness
- Optimistic locking
- Multithreading in EOF
- Concurrent database access
- Manipulating raw rows and SQL



Agenda

- Memory management
- Data freshness
- Optimistic locking
- Multithreading in EOF
- Concurrent database access
- Manipulating raw rows and SQL



Memory Management

- Java garbage collection
- Java references
- Faults
- Deferred faults
- Shared Editing Context

Memory Management

- EOEditingContext uses weak references
- Only inserted, deleted, or updated objects are retained
- Enterprise Objects use strong references back to their EOEditingContext

Memory Management

- registeredObjects() is not especially useful
  - GC giveth, GC taketh away
- setRetainsRegisteredObjects()
- EOSharedEditingContext always uses strong references

Memory Management

Leverage faulting

- sizeof(fault) == sizeof(eo)
- Deferred faults for to-many relationships are memory efficient
- Enterprise Objects extend EOGenericRecord to use deferred faults
- awakeFromFetch & awakeFromInsertion

Memory Management

- Blobs factored into their own table
- Avoid premature optimizations, but...
  - Some to-many relationships **not** class properties
  - Rarely used attributes placed across a relationship
  - Fetch instead of a to-many relationship

Memory Management

EOSharedEditingContext

- Shared is a misnomer; “Mostly Read Only”
- Share memory for entire Enterprise Object, not just reusing the row level snapshot
- No outgoing relationships
- Can only be used with ECs sharing the same EOObjectStoreCoordinator

Memory Management

So, I've got a million EOs and...

- Divide operations into batches
- Raw rows
- Project Wonder
  ERXFetchSpecificationBatchIterator
  ERXEOControlUtilities.objectCountWithQualifier()

Memory Management

- Intercept rows at the adaptor level
- Custom JDBCChannel sublcass overriding fetchRow()
  - Invoke super.fetchRow()
  - Use that row
  - Return null (or the row, as desired)

Agenda

- Memory management
- Data freshness
- Optimistic locking
- Multithreading in EOF
- Concurrent database access
- Manipulating raw rows and SQL

Data Freshness

- EOF is a cache
- Refaulting
- Invalidating
- Fetch time stamps
- EODatabase is the cache object

Data Freshness

- Refaulting affects 1 editing context
  - No notification
  - No affect on other ECs
  - Existing snapshot used when fault is fired
  - No DB round trip

Data Freshness

- Invalidating affects entire stack
  - ObjectsChangedInStore notification to all ECs
  - All ECs lose changes for that object
  - Snapshot is discarded
  - DB round trip when fault is fired

Data Freshness

- If you must invalidate:
  - Firing faults for all invalidated objects could cause thousands of db round trips
  - Batch faulting better
  - Manually fetching to restore snapshots best
    - One fetch specification v.s. one fetch per fault
    - Prefetching keypaths extremely useful

Data Freshness

- refreshObject()
  - Smarter refaultObject()
  - Merges changes for updates
  - Skips inserts
  - Uses existing row snapshot

Data Freshness

- Fetch time stamps do not work as you wish they would
- Fetch time stamp lag is relative
  - Default is 1 hour
- All fetch time stamps are absolute
  - Like June 25th 4:15pm
- Periodically reset fetch time stamp on EC for relative freshness

Data Freshness

- setFetchTimestamp()
  - session.awake()
- EOFetchSpecification.setRefreshesRefetchedObjects()
- Prefetching can refresh a lot of data in one operation

Data Freshness

Manipulating the EODatabase snapshot cache

- Cheating at Faulting
  recordSnapshotForGlobalID(NSDictionary snapshot, EOGlobalID gid)
- Preflighting a to-many relationship for inserted EOs
  recordSnapshotForSourceGlobalID(NSArray gids, EOGlobalID gid, String name)
- Refreshing a to-many relationship snapshot
  recordSnapshotForSourceGlobalID(null, EOGlobalID gid, String name)
- Remember to lock and unlock the EOObjectStoreCoordinator

Agenda

- Memory management
- Data freshness
- Optimistic locking
- Multithreading in EOF
- Concurrent database access
- Manipulating raw rows and SQL

Optimistic Locking

- EOF Developer's Guide
  http://developer.apple.com/techpubs/webobjects/Enterprise_Objects
- Chapter 9 "Update Strategies"
- Dealing with multiple writers affecting the same data

Optimistic Locking

- Cache validity
- EOF is a big cache of database rows
  - Multiple WOA instances
  - Multiple EOObjectStoreCoordinators
  - External writers

Optimistic Locking

- Avoiding the problem
  - Keep data fresh as described earlier
  - Code to propagate distributed notifications

Optimistic Locking

- WireHose Change Notification Framework:
  http://www.wirehose.com/download
  WHChangeNotification
- Project Wonder
  ERChangeNotificationJMS

Optimistic Locking

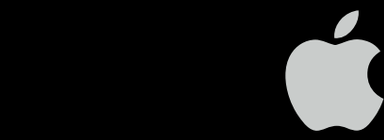
- Recovering from the problem
  - EOAdaptorOperation
    - Attempted row changes
    - Operation type code
  - EODatabaseOperation
    - EOEnterpriseObject
    - Out of sync row snapshot
- Refaulting removes EO from updated/deleted lists
- Cannot refault a newly inserted EO

Optimistic Locking Demo

Ben Trumbull
webobjects@apple.com

Agenda

- Memory management
- Data freshness
- Optimistic locking
- Multithreading in EOF
- Concurrent database access
- Manipulating raw rows and SQL



Multithreading in EOF

- Many good books available
- “Concurrent Programming in Java”
- Necessary concepts:
  - Synchronized methods and blocks
  - Try and finally
  - wait()
  - notify()
- WebObjects 5.2 Delta Documentation

    http://developer.apple.com/techpubs/webobjects/Whatsnew

Multithreading in EOF

- Java apps are always multithreaded
  - Object finalization
  - WOSession time outs
  - Timers
  - Notifications
  - WOLongRequest pages
  - Independent user threads

Multithreading in EOF

- Use try-finally

```
resource.lock();
try {
   /* use resource */
   /* never mess with the lock itself */
} finally {
   resource.unlock();
}
```

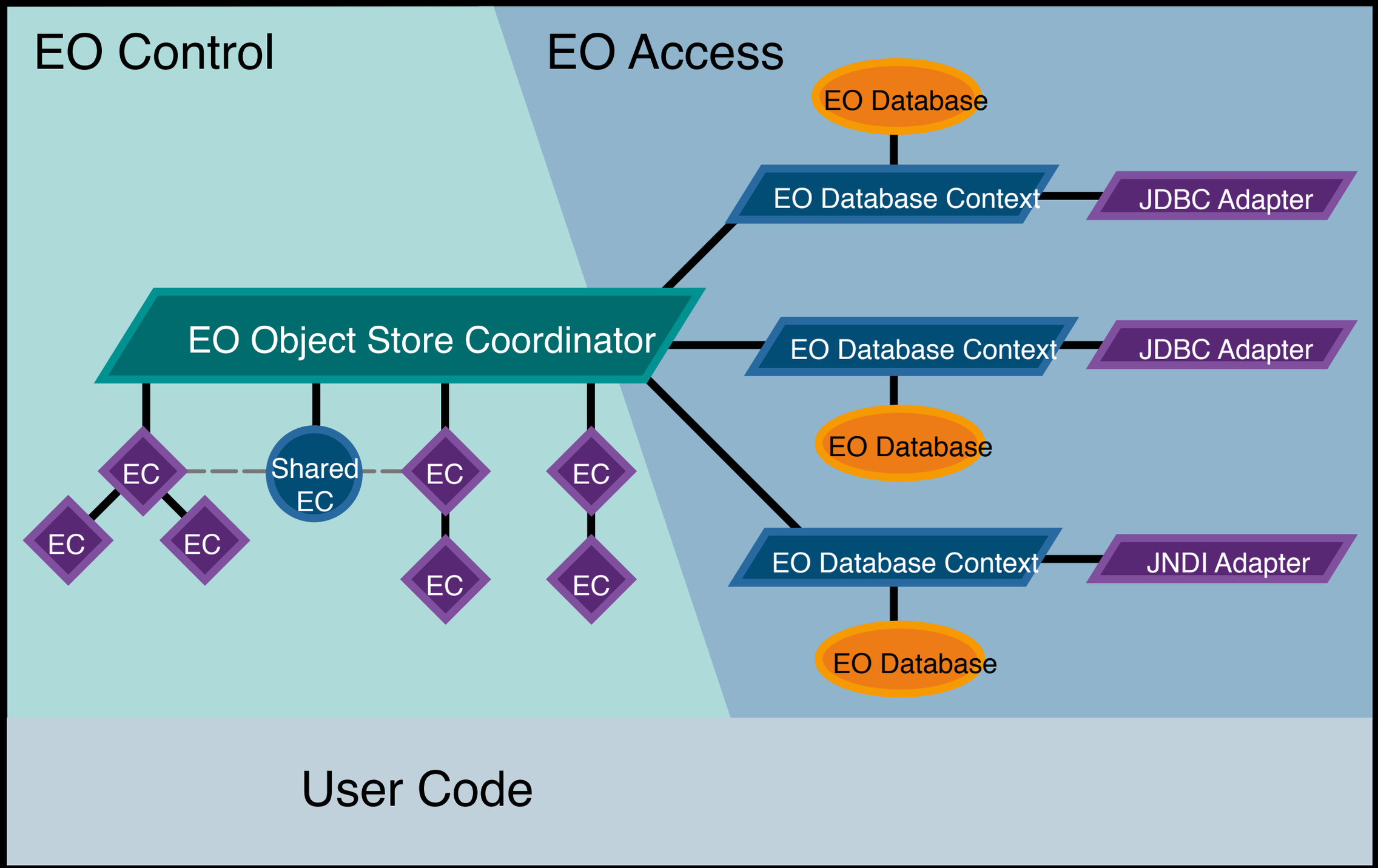
Multithreading in EOF

- Lock objects you access directly
- NSLocking interface
  - Objects that do not implement NSLocking are not thread safe
- tryLock() supported by
  - NSRecursiveLock
  - EOEditingContext
- control-\ or kill -QUIT pid produces stack trace
- OptimizeIt and JProbe

Multithreading in EOF

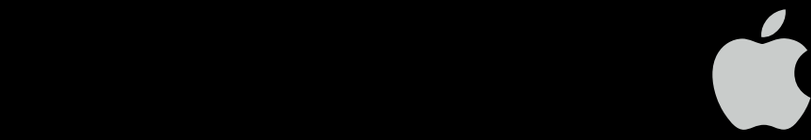
- EOEditingContext
  - Nested ECs share their parent's lock
  - Must still lock nested ECs
- Shared Editing Contexts lock themselves
  - Only EOF object that locks itself
- Must always lock EC to access Enterprise Objects
  - "Read only" operations not thread safe
- EOObjectStoreCoordinators lock all their object stores

Multithreading in EOF



Multithreading in EOF

- Remaining bottlenecks:
  - EOModel
  - EOEntity
  - NSNotificationCenter
- WebObjects–allowsConcurrentRequestHandling
  - Permits dispatching multiple WO requests (sessions, direct actions)
  - Zero affect on EOF



Serving as a Warning to Others

- Non-parallel locks are like debts
- Intersecting locks increase the possibility for deadlock
- Synchronized is its own additional lock
- Blocking operations within a synchronized scope are bad
- Lock for blocking operations with a complete lock object
  - NSRecursiveLock
  - Java's wait()/notify()

Agenda

- Memory management
- Data freshness
- Optimistic locking
- Multithreading in EOF
- Concurrent database access
- Manipulating raw rows and SQL

Concurrent Database Access

- WebObjects 5.2 Delta Documentation

  http://developer.apple.com/techpubs/webobjects/Whatsnew

- Putting multithreading to use
- Simultaneous database operations

Concurrent Database Access

- EOObjectStoreCoordinator is root object in EOF
- EOObjectStoreCoordinator has its own EOAccess stack
  - Locks
  - Database channel
  - Snapshot cache

Concurrent Database Access

- Creating a new database connection is easy

```
 EOObjectStoreCoordinator osc =
new EOObjectStoreCoordinator();

EOEditingContext ec = new EOEditingContext(osc);
```

Concurrent Database Access

- Ideal for raw row fetches
  - No duplicated snapshots
  - No NSNotificationCenter overhead
- Actual concurrency throttled by your database:
  - Isolation level, locking discipline, etc.

Agenda

- Memory management
- Data freshness
- Optimistic locking
- Multithreading in EOF
- Concurrent database access
- Manipulating raw rows and SQL

Raw Rows

- Low level operations
- Result set is an NSArray of NSDictionarys
- No object graph
- No row caching
- Option for direct SQL use

Raw Rows

- EOFetchSpecification.setFetchesRawRows()
- Custom SQL query in fetch spec hints
- EOUtilities.rawRowsForSQL()
- SQL expressions for attribute definitions
- EOEntity external query
- EOAdaptorChannel.evaluateExpression()

Raw Rows

- EOs and dictionaries both implement NSKeyValueCoding
- EOFetchSpecification's CustomQueryExpressionHintKey
  setHints()
- Promoting raw rows into EOs
  EOUtilities.faultWithPrimaryKey() or
  EOEntity.globalIDForRow()
- Demoting EOs into primary keys
  EOEntity.primaryKeyForGlobalID()
- ThinkMoviesX example does it all

Raw Rows

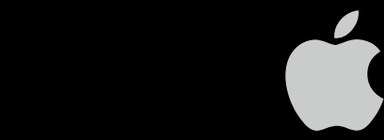
- Features EOF does not provide
  - IN qualification
  - Null relationship qualification
    - Subquery necessary
- Project Wonder

Using Raw SQL Demo

Steve Miner
webobjects@apple.com

What We've Reviewed Today

- Memory management
- Data freshness
- Optimistic locking
- Multithreading in EOF
- Concurrent database access
- Manipulating raw rows and SQL



Upcoming WebObjects 5.2.2

- MacOS X Server and J2EE
  - JBoss Integration
- EOF impact:
  - log4j
    NSLog.Log4JLogger
  - JNDI lookup
    System properties
    JDBC Connection URLs
  - Assorted bug fixes

For More Information

Reference Library

Documentation

WebObjects Developer Documentation

http://developer.apple.com/techpubs/webobjects

EOF Developer's Guide

http://developer.apple.com/techpubs/webobjects/Enterprise_Objects

WebObjects API Reference

http://developer.apple.com/techpubs/Reference/API

WebObjects 5.2 Delta Documentation

http://developer.apple.com/techpubs/webobjects/Whatsnew

Reference Library

Reporting Bugs

http://bugreport.apple.com

Technical Notes

TN2027: How to write a JDBC plugin

http://developer.apple.com/technotes/tn/tn2027.html

Reference Library

Project Wonder

http://wonder.sourceforge.net

Third-Party Sites

http://www.omnigroup.com

http://www.stepwise.com

Who to Contact

Andreas Wendker
Manager, WebObjects Engineering
webobjects@apple.com

Bob Fraser
Senior Product Manager, WebObjects and Java
webobjects@apple.com

Katherine Wenc
Manager, Technical Publications
webobjects@apple.com

Enterprise-Level WebObjects Support And Consulting
AppleCare Enterprise Services
services@apple.com
(800) 630 2775

http://developer.apple.com/wwdc2003/urls.html

Reference Library

Resources

Product Information

http://www.apple.com/webobjects

Developer Support

http://developer.apple.com/webobjects

Enterprise-Level WebObjects Support And Consulting

http://www.apple.com/services/technicalsupport

http://www.apple.com/services/consulting

Mailing List

WebObjects Announcements

webobjects-announce@apple.com

Q&A

**Ben Trumbull, Steve Miner, Brent Shank,
Bill Bumgarner, Andreas Wendker**

webobjects@apple.com

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference